20 VIDEOLEZIONI DI BASIC
PER IMPARARE CON LO SPECTRUM

GRUPPO
EDITORIALE
JACKSON

5

Spectrum
16K/48K/PLUS

VIDEO BASIC SPECTRUM
Pubblicazione quattordicinale
edita dal Gruppo Editoriale Jackson
Direttore Responsabile:
Giampietro Zanga
Direttore e Coordinatore
Editoriale: Roberto Pancaldi
Autore: Softidea - Via Indipendenza 88 - Como
Redazione software:
Francesco Franceschini, Roberto Rossi,
Alberto Parodi, Luca Valnegri
Segretaria di Redazione:
Marta Menegardo
Progetto grafico:
Studio Nuovaidea - Via Longhi 16 - Milano
Impaginazione:
Silvana Corbelli
Illustrazioni:
Cinzia Ferrari, Silvano Scolari
Fotografie:
Marcello Longhini
Distribuzione: SODIP
Via Zuretti, 12 - Milano
Fotocomposizione: Lineacomp S.r.l.
Via Rosellini, 12 - Milano
Stampa: Grafika '78
Via Trieste, 20 - Pioltello (MI)
Direzione e Redazione:
Via Rosellini, 12 - 20124 Milano
Tel. 02/6880951/5

Autorizzazione alla pubblicazione Tribunale di
Milano n° 422 del 22-9-1984
Spedizione in abbonamento postale Gruppo II/70
(autorizzazione della Direzione Provinciale delle
PPTT di Milano).
Prezzo del fascicolo L. 8.000
Abbonamento comprensivo di 5 raccoglitori L. 165.000
I versamenti vanno indirizzati a: Gruppo
Editoriale Jackson S.r.l. - Via Rosellini, 12
20124 Milano, mediante emissione di assegno
bancario o cartolina vaglia oppure
utilizzando il c.c.p. n° 11666203.
I numeri arretrati possono essere
richiesti direttamente all'editore
inviando L. 10.000 cdu. mediante assegno
bancario o vaglia postale o francobolli.
Non vengono effettuate spedizioni contrassegno.

**Gruppo Editoriale
Jackson**

SOMMARIO

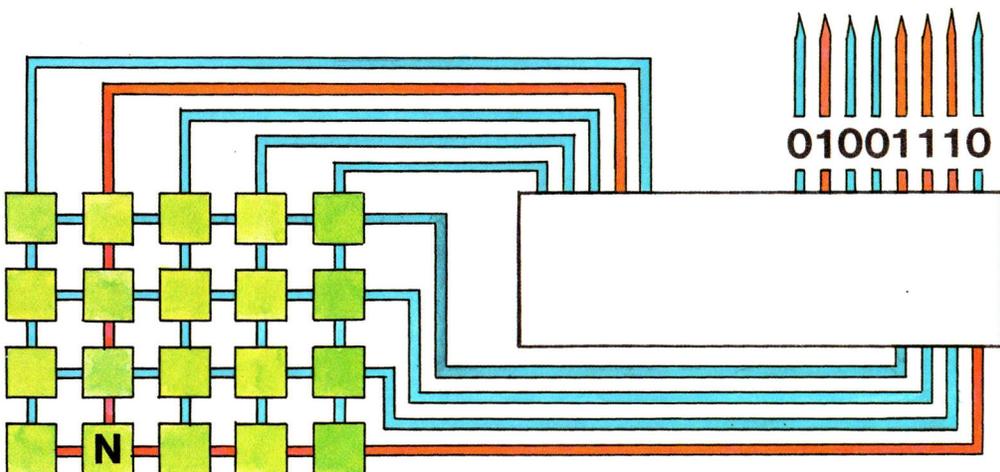
Introduzione

In questa lezione approfondiremo la conoscenza della tastiera, il principale dispositivo di ingresso dei dati.
Non tutte le tastiere sono però uguali i meccanismi variano, infatti, da un tipo all'altro così come le caratteristiche non che il principio di funzionamento.
Indissolubilmente legati alla tastiera sono il codice ASCII e il set dei caratteri.
Impareremo poi, a conoscere e usare, le CODE - CHR$ - INKEY$ e FOR-TO - STEP-NEXT, che ti permetteranno di eseguire quante volte vuoi un gruppo di istruzioni.
Per finire, una tecnica indispensabile a ogni programmatore: i cicli automatici.

Schema e funzionamento dei tipi di tastiera

La tastiera costituisce sicuramente il principale dispositivo di ingresso delle informazioni di cui è fornito il tuo calcolatore.
È fondamentalmente attraverso essa che ti è infatti possibile comunicare all'unità centrale tutti i comandi, le istruzioni ed i dati che intendi eseguire o memorizzare.
Un computer senza tastiera è come una automobile senza volante: lo potresti mettere in moto ed arrestare, ma non controllare ed utilizzare.
È pertanto importante che tu capisca, al di là del semplice e consueto utilizzo di tutti i giorni, la struttura ed il principio di funzionamento della tastiera di un elaboratore.
Prima di affrontare il discorso è però necessario precisare e chiarire con esattezza cosa si intende con il termine «tastiera». Tale parola specifica infatti solo ed esclusivamente il dispositivo utilizzato per l'ingresso dei dati.
Chiamare «tastiera» un intero calcolatore (come fanno alcune persone non molto informate) è assolutamente sbagliato e scorretto!
Come hai potuto verificare, la tastiera del tuo Spectrum è sostanzialmente identica, nell'aspetto e nel funzionamento, a quella di una comune macchina da scrivere: basta schiacciare il tasto corrispondente al carattere prescelto ed il gioco è fatto.
Eventualmente, mediante la combinazione di due o tre tasti premuti in contemporanea, si possono comporre ulteriori lettere, simboli o comandi che di solito non sono disponibili sulle normali macchine da scrivere.
Una particolarità che forse ti è sfuggita è la disposizione delle lettere: esse sono infatti ordinate secondo lo standard statunitense, chiamato QWERTY.
Questo nome, assegnato alle tastiere di tipo americano, nasce proprio dalla collocazione dei primi sei tasti alfabetici della seconda fila di tasti.
Nelle tastiere cosiddette europee, invece, la Z è in seconda posizione al posto della W; da qui il nome QZERTY.
Ulteriori differenze sono la posizione della M e la disposizione di quasi tutti i simboli e segni di punteggiatura.
Nulla cambia comunque agli effetti pratici: entrambe le tastiere (americana ed europea) si comportano in modo assolutamente identico ed affidabile ai fini del funzionamento.
Cerchiamo ora di capire

come effettivamente funziona una tastiera, cioè cosa succede quando premi un tasto del tuo Spectrum.
Tutti i tasti presenti sulla tastiera sono connessi elettricamente (cioè tramite fili conduttori di elettricità) ad un particolare circuito integrato, che ne rileva l'effettivo azionamento e produce per ciascuno dei tasti premuti un unico e distinto codice numerico binario a 8 bit.
L'unità centrale, quando riceve una simile combinazione, è così immediatamente in grado (normalmente per mezzo di un programma memorizzato su ROM o di un ulteriore circuito elettronico) di distinguere ed individuare il particolare carattere premuto, in modo da eseguire le operazioni richieste.
Così, per esempio, quando premi la lettera A, all'uscita del circuito di codificazione (questo è il termine tecnico usato per indicare tale componente) compare il codice binario 01000001, corrispondente a 65 in decimale. A tale codice (e solo ad esso!) corrisponde per la CPU il carattere A: non vi è quindi alcuna possibilità che nella macchina insorgano confusioni ed errori.

Un circuito integrato individua e riconosce il tasto premuto ed emette il codice binario corrispondente.

Il codice ASCII

I codici numerici da assegnare a ciascuno dei caratteri più utilizzati non vengono scelti arbitrariamente dalla casa costruttrice, ma sono il frutto di una cooperazione avvenuta tra utenti di apparecchiature ed industrie operanti nel ramo della elaborazione dei dati. In origine tali codici erano stati infatti realizzati con lo scopo di semplificare e standardizzare le comunicazioni tra i diversi calcolatori, eliminando così tutti i problemi connessi a differenti rappresentazioni dei dati o delle informazioni.
La diffusione sempre più ampia dei personal computer ha fatto sì che

Decimale	ASCII	Decimale	ASCII	Decimale	ASCII
0	NUL	43	+	86	V
1	SOH	44	,	87	W
2	STX	45	−	88	X
3	ETX	46	.	89	Y
4	EOT	47	/	90	Z
5	ENQ	48	0	91	[
6	ACK	49	1	92	\
7	BEL	50	2	93	]
8	BS	51	3	94	∧
9	HT	52	4	95	—
10	LF	53	5	96	`
11	VT	54	6	97	a
12	FF	55	7	98	b
13	CR	56	8	99	c
14	SO	57	9	100	d
15	SI	58	:	101	e
16	DLE	59	;	102	f
17	DC1	60	<	103	g
18	DC2	61	=	104	h
19	DC3	62	>	105	i
20	DC4	63	?	106	j
21	NAK	64	@	107	k
22	SYN	65	A	108	l
23	ETB	66	B	109	m
24	CAN	67	C	110	n
25	EM	68	D	111	o
26	SUB	69	E	112	p
27	ESC	70	F	113	q
28	FS	71	G	114	r
29	GS	72	H	115	s
30	RS	73	I	116	t
31	US	74	J	117	u
32	spazio	75	K	118	v
33	!	76	L	119	w
34	"	77	M	120	x
35	#	78	N	121	y
36	$	79	O	122	z
37	%	80	P	123	{
38	&	81	Q	124	|
39	'	82	R	125	}
40	(	83	S	126	~
41	)	84	T	127	DEL
42	*	85	U		

tale codificazione, chiamata ASCII (abbreviazione di American Standard Codes for Information Interchange, cioè codici standard americani per l'interscambio delle informazioni), sia diventata di fatto uno "standard" presente nella totalità dei computer. I caratteri alfabetici e di punteggiatura presenti sul tuo Spectrum, sono quindi composti con le stesse combinazioni di bit codificate negli altri calcolatori quando su di essi vengono premuti i tasti corrispondenti.

Tasti e tastiere

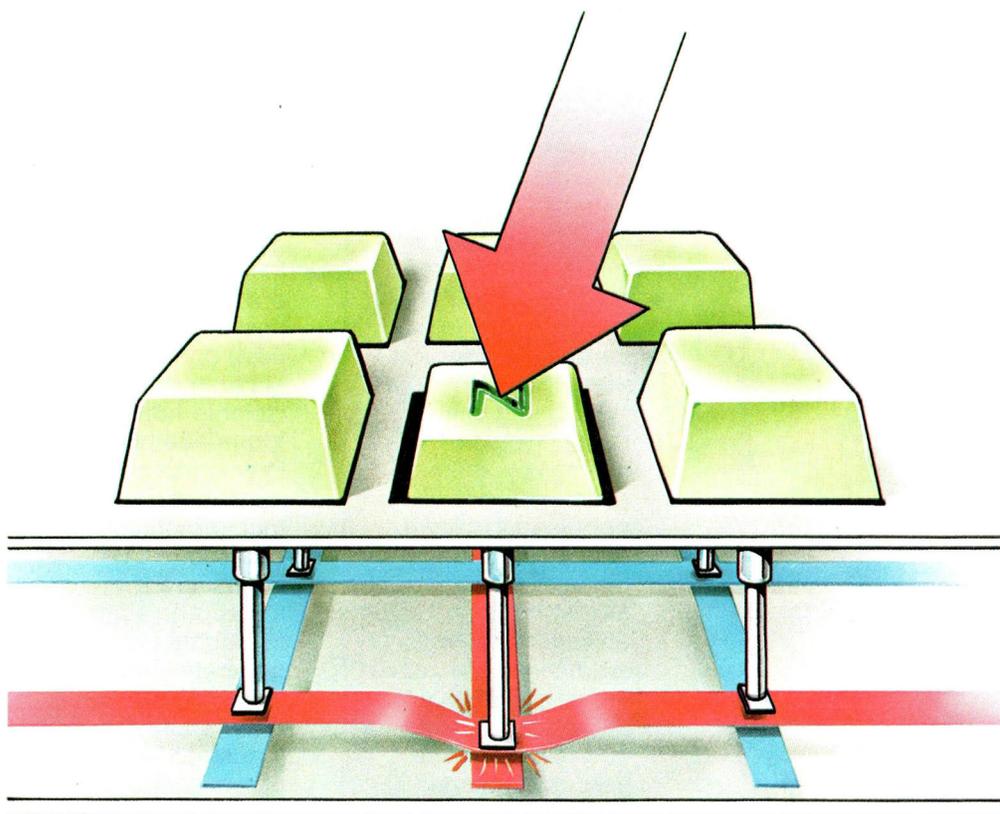
La tastiera è sottoposta a continue sollecitazioni meccaniche. La sua vita (o durata) dipende in gran parte dalla qualità dei tasti: può andare da qualche decina di migliaia di battute (nelle tastiere veramente scadenti) a molte decine di milioni.
Vediamo succintamente i vari tipi di tasti, a partire dai migliori:

- **tasti ad effetto Hall**: sfruttano l'effetto di un magnetino mobile sulla corrente che attraversa un semiconduttore. Avendo pochissima meccanica, la vita media si misura in miliardi di battute;
- **tasti capacitivi**: sono condensatori la cui capacità varia premendo il tasto. Hanno una vita di molte decine di milioni di battute e sono usati nei migliori personal computer;
- **tasti a reed**: un contatto posto all'interno di un'ampolla di vetro (il reed) viene chiuso da un magnetino montato sul nastro. La vita è di qualche decina di milioni di battute. A causa della concorrenza dei più robusti tasti capacitivi, il loro uso si è molto ridotto negli ultimi anni;
- **tasti meccanici standard**: sono impiegati da molti personal computer. La loro vita dipende dalla qualità costruttiva. Nel caso migliore è di alcune decine di milioni di battute. Sono sensibili alle condizioni ambientali (umidità, polvere);
- **tasti a bolla**: sono quelli usati nelle calcolatrici tascabili. Una bolla di metallo si rovescia sotto la pressione del tasto. La loro vita è in genere limitata;
- **tastiere a membrana o a film**: i tasti sono costituiti da due fogli (film) conduttori, tesi e separati da un foglio isolante forato nella forma adatta. Premendo il foglio superiore i due fogli si toccano chiudendo il contatto. Le tastiere a membrana sono impiegate generalmente in personal economici od in applicazioni industriali, in quanto garantiscono un ottimo isolamento dai

fattori esterni che possono pregiudicarne il funzionamento. La durata dipende dalla costruzione.

A parte le differenze costruttive, comunque, possiamo ricondurre a una sola sequenza le operazioni che esegue il tuo computer, o meglio la CPU per analizzare la tastiera e individuare il tasto premuto.

Vediamo in modo estremamente semplificato cosa accade:

1. periodicamente, e ad intervalli stabiliti, la CPU interrompe quello che sta facendo per rivolgere la sua attenzione alla tastiera;
2. esegue la cosiddetta routine di interrupt memorizzata nella ROM;
3. controlla (in gergo, scanning) lo stato della tastiera, cioè se un tasto è stato premuto;
4. se questo è avvenuto, la CPU ricava la posizione del tasto stesso;
5. una volta che ha verificato il punto 4 ricontrolla una frazione dopo (ricorda che la velocità delle CPU è dell'ordine dei

megahertz, cioè milioni di volte al secondo), in modo da escludere eventuali interferenze;
6. a questo punto la CPU ritorna al suo lavoro, per riprendere poi la sequenza illustrata.

Per concludere. La tastiera è il principale mezzo di comunicazione tra te e il computer: trattala con cura.
Essendo infatti meccanica, è soggetta ad usura e la durata della sua vita dipende dall'uso che ne fai.
Evita perciò colpi bruschi, pressioni esagerate dei tasti e soprattutto ... quando un programma "non gira" evita di sfogare sulla tastiera la tua rabbia.

Il set dei caratteri

Come ben sai (scusa la pedanteria, ma occorre) il tuo Spectrum come qualsiasi altro computer conosce, sa usare e memorizzare solo numeri, per di più solo numeri binari.
Come fa allora a capire e visualizzare sul video il punto di domanda (?), la scritta "ciao", l'asterisco (*)?
Semplice. Il tuo Spectrum trasforma in numeri tutti i caratteri alfabetici, numerici e speciali (il set di caratteri della macchina) che è in grado di gestire, inviare e ricevere.
Per questa codifica utilizza un codice assai simile usato da tutti gli altri personal computer: il citato codice ASCII.
È per questo che il tuo computer sa "manipolare" anche stringhe: le interpreta come una successione di numeri codificati corrispondenti ad una precisa tabella di caratteri presente nella memoria.
Quello che è importante è che a ogni codice corrisponde uno e un solo carattere, in modo che il computer non abbia mai alcuna ambiguità di interpretazione.
Come l'ASCII anche il set dei caratteri del tuo Spectrum è un codice a 7 bit; sono perciò possibili 128 (cioè 2^7) combinazioni, alle quali corrispondono altrettanti caratteri. Molti di questi caratteri sono immediatamente e facilmente riconoscibili anche alla prima occhiata, essendo di frequente utilizzo per chiunque: lettere alfabetiche, caratteri di punteggiatura, cifre numeriche.
Altri invece, prima di assumere un qualsiasi significato, richiedono un attimo di riflessione; altri ancora sono assolutamente estranei

ai simboli generalmente usati dall'uomo. Questi ultimi sono i cosiddetti caratteri speciali (o caratteri di controllo): tramite essi puoi infatti impartire dei particolari comandi, che vanno ad adattarsi alle caratteristiche di struttura e di funzionamento della macchina.
Pur non avendo corrispettivo nel linguaggio umano, i caratteri di controllo sono importantissimi per il tuo Spectrum: è grazie ad essi che puoi, per

Codice	*Carattere*	*Codice*	*Carattere*	*Codice*	*Carattere*
0		44	,	91	[
1		45	—	92	/
2		46	.	93	]
3	non usati (0–5)	47	/	94	↑
4		48	0	95	_
5		49	1	96	£
6	virgola della **PRINT**	50	2	97	a
7	**EDIT**	51	3	98	b
8	cursore a sinistra	52	4	99	c
9	cursore a destra	53	5	100	d
10	cursore su	54	6	101	e
11	cursore giù	55	7	102	f
12	**DELETE**	56	8	103	g
13	**ENTER**	57	9	104	h
14	numero	58	:	105	i
15	non usato	59	;	106	j
16	car. controllo **INK**	60	<	107	k
17	car. controllo **PAPER**	61	=	108	l
18	car. controllo **FLASH**	62	>	109	m
19	car. controllo **BRIGHT**	63	?	110	n
20	car. controllo **INVERSE**	64	@	111	o
21	car. controllo **INVERSE**	65	A	112	p
22	car. controllo **OVER**	66	B	113	q
23	car. controllo **AT**	67	C	114	r
24	car. controllo **TAB**	68	D	115	s
25		69	E	116	t
26		70	F	117	u
27		71	G	118	v
28	non usati (25–31)	72	H	119	w
29		73	I	120	x
30		74	J	121	y
31		75	K	122	z
32	spazio	76	L	123	{
33	!	77	M	124	|
34	”	78	N	125	}
35	#	79	O	126	~
36	$	80	P	127	©
37	%	81	Q	128	□
38	&	82	R	129	▝
39	’	83	S	130	▘
40	(	84	T	131	▀
41	)	85	U	132	▗
42	*	86	V	133	▐
43	+	87	W	134	▚
		88	X	135	▜
		89	Y	136	▖
		90	Z	137	▞

Codice	*Carattere*		*Codice*	*Carattere*	*Codice*	*Carattere*
138	▌		185	**EXP**	232	**CONTINUE**
139	▛		186	**INT**	233	**DIM**
140	▄		187	**SQR**	234	**REM**
141	▟		188	**SGN**	235	**FOR**
142	▙		189	**ABS**	236	**GO TO**
143	█		190	**PEEK**	237	**GO SUB**
144	(a)	caratteri grafici definiti dall'utente	191	**IN**	238	**INPUT**
145	(b)		192	**USR**	239	**LOAD**
146	(c)		193	**STR$**	240	**LIST**
147	(d)		194	**CHR$**	241	**LET**
148	(e)		195	**NOT**	242	**PAUSE**
149	(f)	caratteri grafici definiti dall'utente	196	**BIN**	243	**NEXT**
150	(g)		197	**OR**	244	**POKE**
151	(h)		198	**AND**	245	**PRINT**
152	(i)		199	<=	246	**PLOT**
153	(j)		200	>=	247	**RUN**
154	(k)		201	<>	248	**SAVE**
155	(l)		202	**LINE**	249	**RANDOMIZE**
156	(m)		203	**THEN**	250	**IF**
157	(n)		204	**TO**	251	**CLS**
158	(o)		205	**STEP**	252	**DRAW**
159	(p)		206	**DEF FN**	253	**CLEAR**
160	(q)		207	**CAT**	254	**RETURN**
161	(r)		208	**FORMAT**	255	**COPY**
162	(s)		209	**MOVE**		
163	(t)		210	**ERASE**		
164	(u)		211	**OPEN#**		
165	**RND**		212	**CLOSE#**		
166	**INKEY$**		213	**MERGE**		
167	**PI**		214	**VERIFY**		
168	**FN**		215	**BEEP**		
169	**POINT**		216	**CIRCLE**		
170	**SCREEN$**		217	**INK**		
171	**ATTR**		218	**PAPER**		
172	**AT**		219	**FLASH**		
173	**TAB**		220	**BRIGHT**		
174	**VAL$**		221	**INVERSE**		
175	**CODE**		222	**OVER**		
176	**VAL**		223	**OUT**		
177	**LEN**		224	**LPRINT**		
178	**SIN**		225	**LLIST**		
179	**COS**		226	**STOP**		
180	**TAN**		227	**READ**		
181	**ASN**		228	**DATA**		
182	**ACS**		229	**RESTORE**		
183	**ATN**		230	**NEW**		
184	**LN**		231	**BORDER**		

esempio, indicare il termine di una linea di programma (con il tasto ENTER, spostare il cursore in qualunque punto dello schermo (con le frecce, →, ←) o cancellare un carattere dallo schermo (DELETE).
Abbiamo detto che il set dei caratteri fa uso di codici a 7 bit: il tuo Spectrum è però un elaboratore a 8 bit. Le combinazioni possibili sono di conseguenza 256 (cioè 2^8).
Perché sprecare una simile opportunità? Le restanti 256 − 128 = 128 combinazioni sono state perciò utilizzate dalla Sinclair per definire altri caratteri non appartenenti allo standard ASCII, ma propri dello Spectrum: per esempio i caratteri grafici. Il loro uso è esclusivamente riservato al tuo computer; i calcolatori delle altre marche, non disponendo del corrispondente carattere, non sono infatti nelle condizioni di riconoscerli e quindi di utilizzarli.
È bene che tu tenga presente questa limitazione, se vuoi scrivere programmi "portatili", che possano cioè girare anche su altre macchine.

CODE

A volte può essere utile conoscere il codice ASCII di un dato carattere o, viceversa, essere in grado di produrre o stampare un certo carattere dato il suo codice. Il BASIC mette a tua disposizione due comandi, attraverso i quali puoi convertire i caratteri in codici ed i codici in caratteri: CODE e CHR$. Entrambi questi comandi sono delle funzioni; è quindi necessario che a ciascuno di essi tu fornisca un argomento sul quale operare.
CODE produce il valore del codice ASCII, cioè un numero corrispondente al primo carattere di una stringa. L'argomento deve pertanto essere una stringa sotto forma di costante (racchiusa naturalmente tra virgolette) o di variabile. CODE fornisce come risultato un numero compreso tra 0 e 255: i caratteri disponibili sul tuo computer sono infatti, come detto, complessivamente 256. Se l'argomento è una stringa nulla, CODE restituisce il valore 0. Vediamo qualche esempio:

```
PRINT CODE ("A")
```

ottieni 65 codice numerico ASCII del carattere "A"

```
PRINT CODE ("ABCD")
```

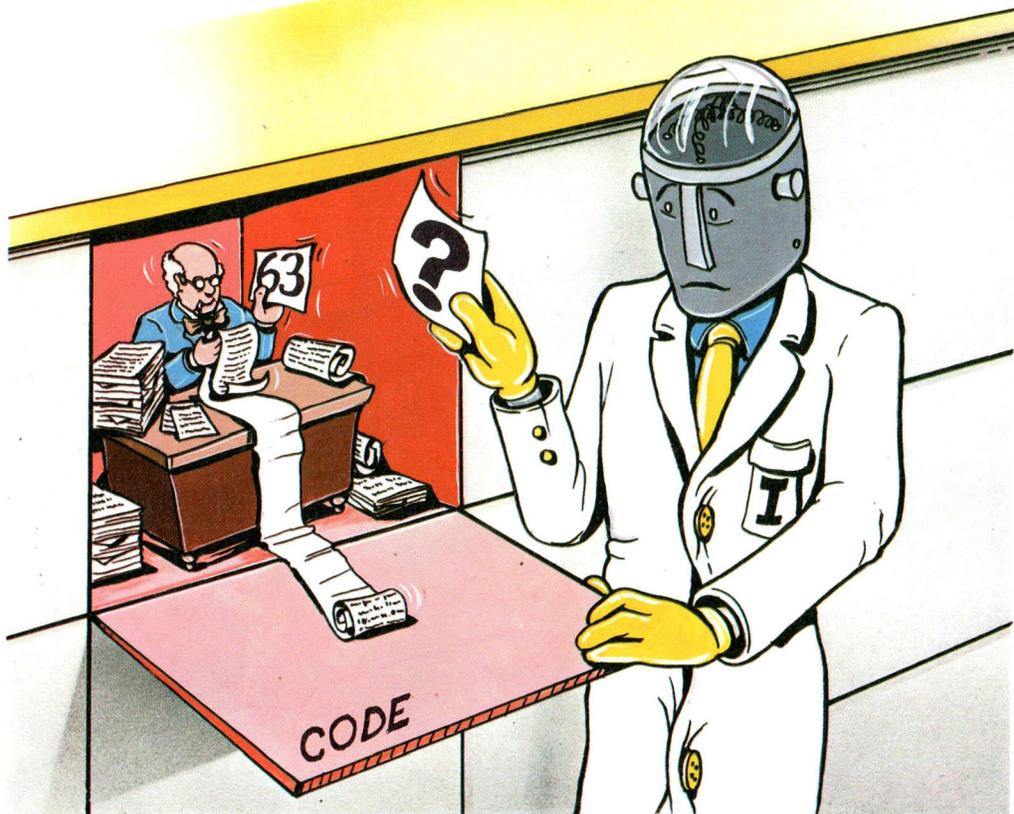
ottieni 65: il primo carattere dell'argomento è "A"

```
50 C$ = "TAVOLO"
60 PRINT CODE (C$)
```

ottieni 84 codice numerico del carattere "T"

Sintassi dell'istruzione

CODE (stringa)

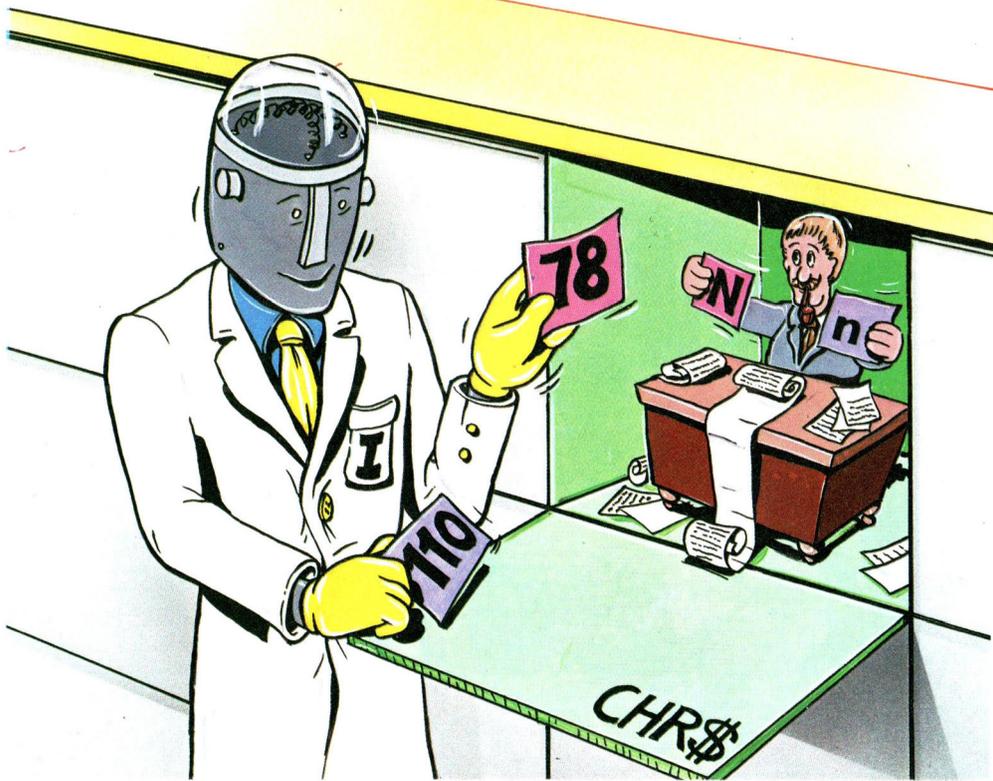
L'argomento di CODE è un carattere. Il risultato è il suo codice numerico.

CHR$

CHR$ è, in un certo senso, la funzione opposta di CODE: essa ti restituisce il carattere corrispondente al numero che avrai utilizzato come argomento.

Tale numero può essere specificato sia per mezzo di una variabile che di un'espressione. L'argomento deve essere compreso tra 0 e 255; in caso contrario l'elaboratore visualizzerà il messaggio di errore INTEGER OUT OF RANGE.
Quando il valore dell'argomento è un numero decimale, CHR$ lo tronca al valore intero immediatamente inferiore:

```
15 PRINT CHR$(151/2)
```

151/2 = 75.5: il numero intero immediatamente più piccolo è 75. A tale argomento corrisponde il carattere K.
Il seguente programma stampa l'intero set dei caratteri disponibili sul tuo Spectrum:

```
10 FOR A = 0 TO 255: PRINT CHR$ (A): NEXT A
```

Alcuni di questi caratteri (i cosiddetti caratteri di controllo), pur facendo parte del set, non sono visualizzabili: essi pertanto non compariranno sullo schermo.
Le funzioni CODE e CHR$ possono essere utilmente impiegate quando, per esempio, si vuole trasformare una lettera maiuscola nel corrispondente carattere minuscolo (o viceversa). Guardando la tabella ASCII puoi infatti immediatamente renderti conto di come il codice di una lettera minuscola sia pari al codice della rispettiva lettera maiuscola aumentato di 32.

L'argomento di CHR$ è un codice numerico, il suo risultato è il carattere corrispondente.

Il programma che segue sfrutta proprio questa caratteristica per stampare in minuscolo il carattere corrispondente alla lettera maiuscola battuta sulla tastiera.

```
10 CLS
20 INPUT A$: IF A$ = "*" THEN GOTO 90: REM
   attende la pressione di un tasto
30 IF CODE(A$) < 65 THEN GOTO 20
40 IF CODE A$ > 90 THEN IF CODE A$ < 97
   THEN GOTO 20
50 IF CODE A$ > 122 THEN GOTO 20: REM
   questa concatenazione di IF fa sì che vengano
   accettati soltanto i caratteri corrispondenti alle
   lettere maiuscole (tra 65 e 90) e minuscole (tra
   97 e 122), ignorando gli altri.
60 IF CODE(A$)< 91 THEN PRINT A$, CHR$
   (CODE (A$) +32) : REM se il carattere è
   maiuscolo, allora stampa anche il
   corrispondente minuscolo.
70 IF CODE (A$) > 96 THEN PRINT A$, CHR$
   (CODE (A$) - 32) : REM se invece è
   minuscolo, allora stampa il corrispondente
   maiuscolo.
80 GOTO 20
90 REM FINE
```

Esempi

```
PRINT CHR$(65)
```

Il codice 65 corrisponde al carattere "A".

```
PRINT CHR$ (18 * 5)
```

L'argomento della funzione CHR$ può essere una espressione il cui valore è compreso tra 0 e 255.

```
S=(100 - 1)
PRINT CHR$ (S/3)
```

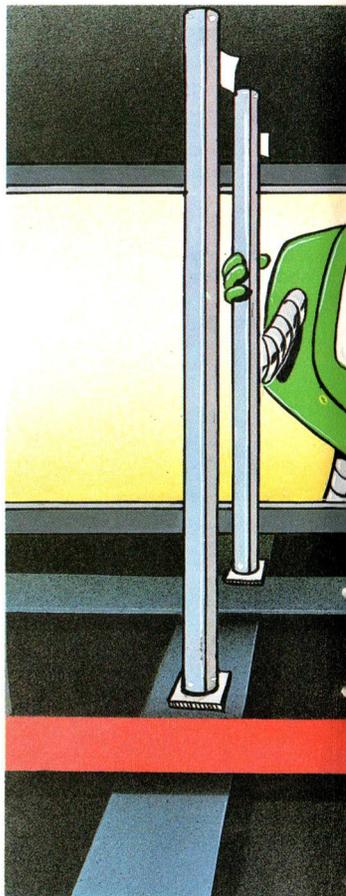
Se vai a vedere la tabella ASCII pubblicata qualche pagina indietro vedrai che al codice 33 corrisponde il punto esclamativo (!).

```
PRINT CHR$ (65.91)
```

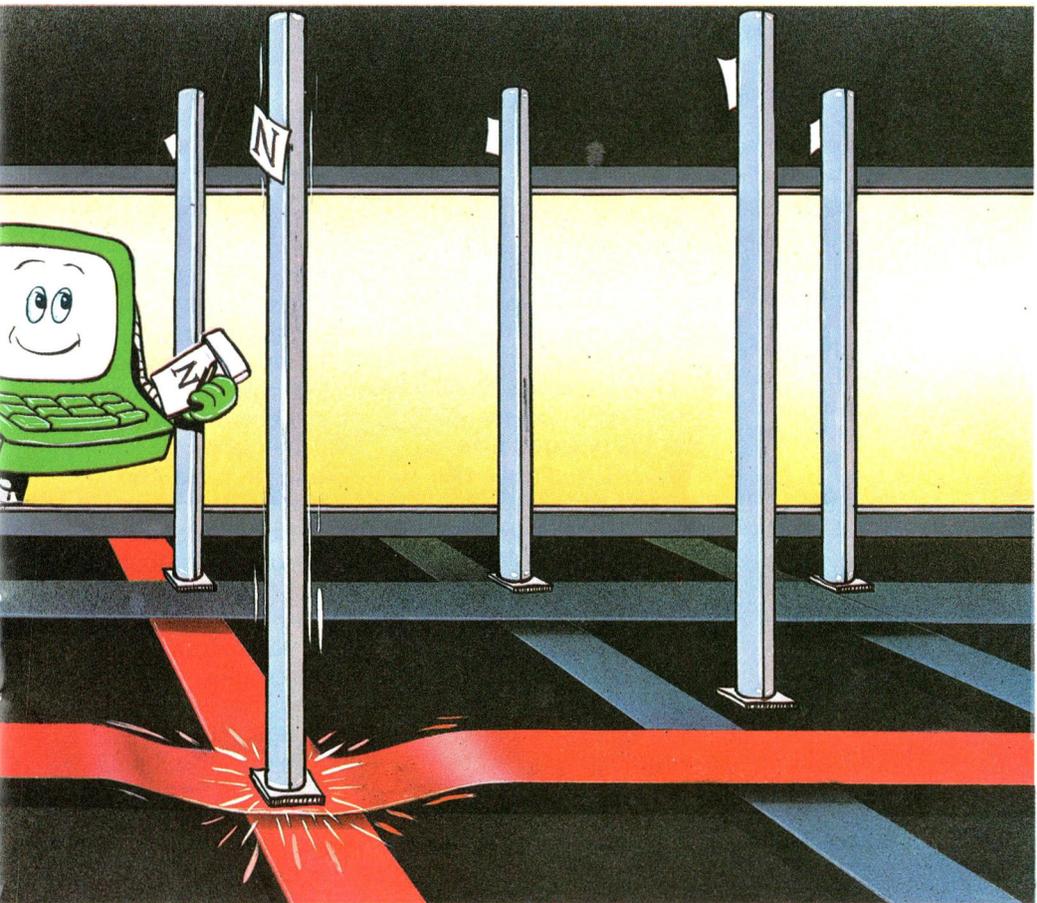
Quando l'argomento di CHR$ è un numero decimale, viene troncato al valore intero immediatamente più vicino, in questo caso 66.
Si ottiene perciò la stampa del carattere B.

Sintassi dell'istruzione

CHR$ (numero)
dove NUMERO può essere un numero una variabile od un'espressione numerica

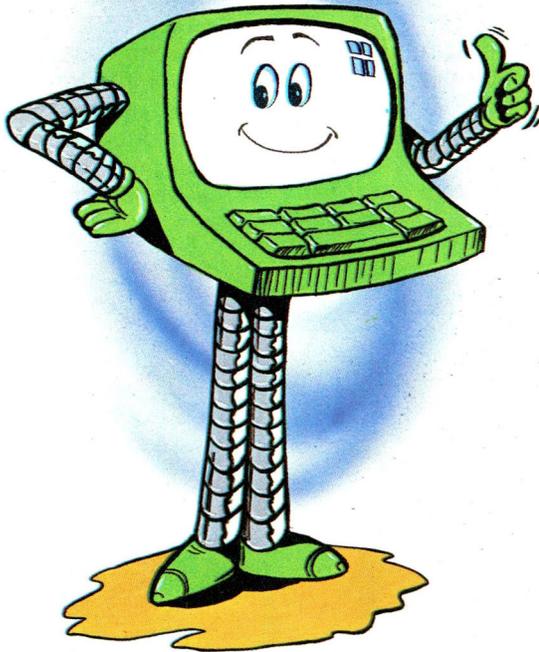
INKEY$

LINGUAGGIO

La funzione INKEY$ consente l'ingresso dalla tastiera del tuo Spectrum di un singolo carattere, senza però dover battere anche il tasto ENTER.
Il carattere corrispondente al tasto premuto non viene inoltre visualizzato sullo schermo.
Perché non usare il già familiare e ormai conosciuto INPUT, allora?
Non è forse vero che anche con INPUT si può introdurre un solo carattere? La differenza è sottile, ma importante.
Come visto parlando di INPUT, se sbagli il dato da introdurre, le conseguenze possono essere catastrofiche i risultati sbagliati o, peggio, programma bloccato.
Con INKEY$, nulla di tutto questo: è quasi

Quando incontra INKEY$ il tuo Spectrum annota il carattere che stai digitando in quel momento.

fatto apposta per permetterti di sbagliare (input controllato) senza che succeda nulla. Anzi, il tuo Spectrum attende tranquillo che tu introduca esattamente la risposta che si aspetta da te.

Il termine "attende" merita una considerazione a parte. Con INPUT il programma viene interrotto per aspettare i dati in ingresso, INKEY$ invece viene eseguito come una normale funzione BASIC, alla velocità di cui è capace l'interprete e quindi ben superiore a quella di qualsiasi uomo. Occorre perciò, inserire INKEY$ in un ciclo di attesa, che renda "umano" il tempo di risposta.
Se vuoi, puoi scoprire da solo (senza leggere il listato si intende) quando i programmi, ad esempio quelli di VIDEOBASIC, fanno uso della funzione INKEY$ o di un INPUT. Prova a pensarci ... Ogni qualvolta il programma richiede la pressione di un tasto seguito da ENTER, vuol dire che sotto c'è INPUT. Altrimenti ... L'argomento di INKEY$ può essere una o più variabili stringa. Quindi, se vuoi che il carattere riconosciuto sia 2 oppure * oppure ! è fondamentale porlo tra virgolette.
Ricapitolando INKEY$ viene solitamente utilizzata per attendere e verificare l'entrata di un qualsiasi carattere dalla tastiera o per attendere che l'utente prema un tasto.
L'istruzione INKEY$ ritorna infatti il carattere premuto sulla tastiera al momento della sua chiamata; quando non è premuto (o non è stato premuto) alcun tasto, INKEY$ assegna alla variabile un valore nullo, ed il programma prosegue normalmente.
L'esempio che segue potrà chiarirti le idee:

```
10 LET A$ = INKEY$
20 IF A$ = " " THEN 10
30 PRINT A$
```

La linea 10 assegna alla variabile A$ il valore del tasto che avrai premuto sul tuo Spectrum; se nessun tasto sarà stato battuto, A$ assumerà valore nullo.
Sia in un caso che nell'altro il programma non subirà alcuna interruzione o sosta.
La riga 20 controlla quindi il valore di A$: se tale variabile conterrà il valore nullo, l'esecuzione riprenderà alla riga 10, ricominciando il ciclo.
L'unica maniera per terminare il programma sarà perciò quella di premere un tasto qualsiasi. Così facendo

comparirà inoltre sullo schermo il carattere battuto (linea 30).
Un tipico utilizzo di INKEY$ è possibile trovarlo in quei programmi che sottopongono l'utente a scelte del tipo:

```
VUOI CONTINUARE? (S/N)
```

Simili domande richiedono come risposta la semplice pressione dei tasti S o N: costruendo una struttura a ciclo, simile a quella illustrata in precedenza, sarà allora possibile scartare automaticamente tutte le risposte che non rientrano tra quelle ammissibili, evitando magari di riempire lo schermo con inutili e antiestetici caratteri:

```
10 CLS
20 PRINT "VUOI CONTINUARE? (S/N)"
30 LET R$ = INKEY$
40 IF R$ <> "S" THEN IF R$ <> "N" THEN
   GOTO 30
50 PRINT R$
60 IF R$ = "S" THEN GOTO 20
70 REM FINE
```

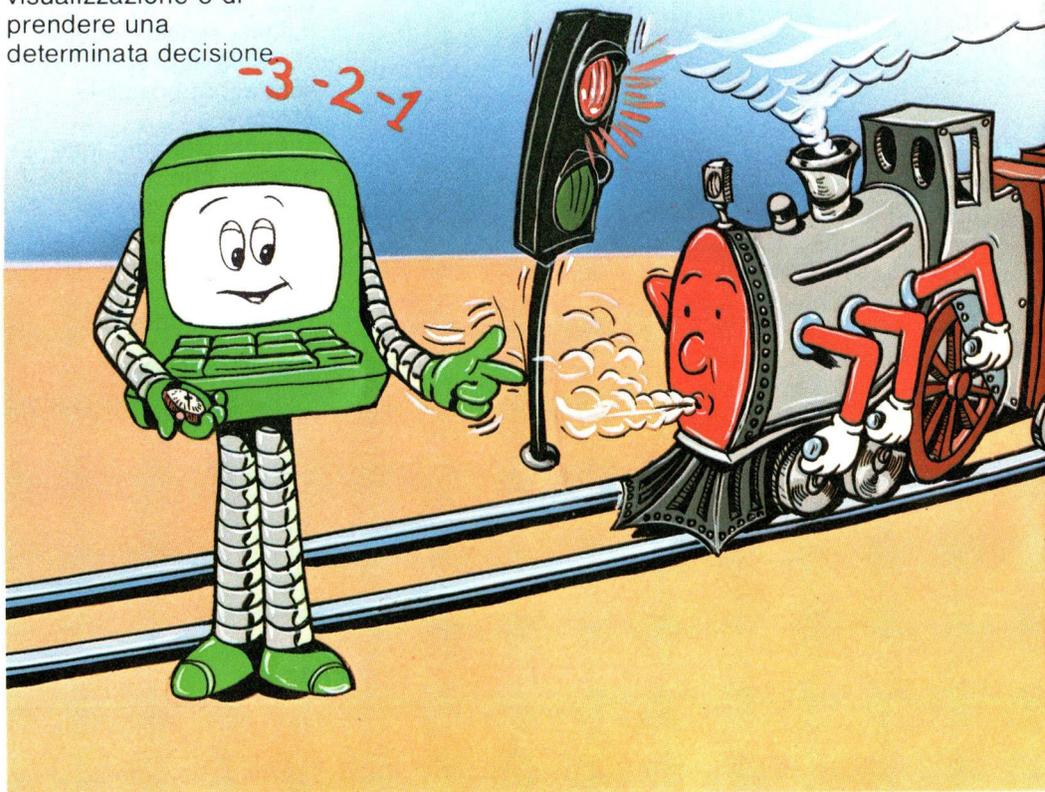
Sintassi della funzione

VARIABILE STRINGA = INKEY$
Come puoi notare, INKEY$ è priva di argomento.

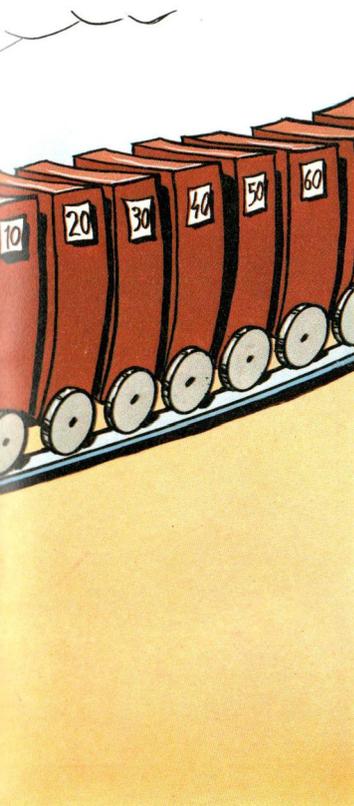
PAUSE

Talvolta, nel corso di un programma, può essere utile avere la possibilità di arrestare momentaneamente l'esecuzione delle varie istruzioni.
In alcuni casi possono infatti rendersi necessarie delle pause che permettano all'utente di leggere e valutare una certa visualizzazione o di prendere una determinata decisione.

Per questo scopo il tuo Sinclair dispone di un particolare comando: PAUSE.
PAUSE n arresta l'esecuzione del programma per un tempo tanto più lungo quanto maggiore è il valore numerico assegnato ad n, al massimo 65535, valore a cui corrisponde una sosta di circa 22 minuti.

Se poni n = 0, il programma si arresterà fino a quando premerai un tasto.
Valori intermedi, compresi cioè tra il valore 0 e 65535, fermeranno l'esecuzione per tempi inferiori. Ad esempio, per n = 50 la pausa è di un secondo; dunque per una pausa di 5 secondi occorre porre n = 250.

PAUSE gode inoltre di un'ulteriore particolarità: se nel corso della sosta viene premuto un tasto qualunque, il programma riprende immediatamente a lavorare, abbreviando la pausa prevista in origine. Questo comando può essere quindi utilmente adoperato per adeguare il tempo di permanenza sullo schermo delle varie visualizzazioni alle capacità di lettura dell'utente del programma.

Sintassi del comando

PAUSE n
dove n è un valore numerico compreso tra 0 e 65535.

FOR, TO, STEP, NEXT

Accade spesso che un programma richieda di eseguire più volte un'istruzione od un gruppo di istruzioni. Supponi, a titolo di esempio, di voler scrivere un programma che moltiplichi la variabile A per i valori 1, 2, 3, 4, e 5. Una possibile soluzione potrebbe essere:

```
10 LET I = 1 : REM I è
   il numero
   da moltiplicare
   per A
20 LET A = A * I
30 LET I = I + 1
40 IF I < 6 GOTO 20
```

Si tratta di un tipico esempio di ciclo, cioè una sequenza di istruzioni eseguita un certo numero di volte. Esiste in BASIC una istruzione particolare che ti permette di realizzare i cicli senza ricorrere a strutture complesse. L'istruzione è composta dalle parole FOR...NEXT.
Immagina di dover ripetere una serie di istruzioni per un determinato numero di volte, e precisamente finché un certo contatore C (avente inizialmente il valore S) raggiunga il valore A. Utilizzando l'istruzione FOR...NEXT potrai scrivere:

```
FOR C = S TO A
```

fai qualcosa

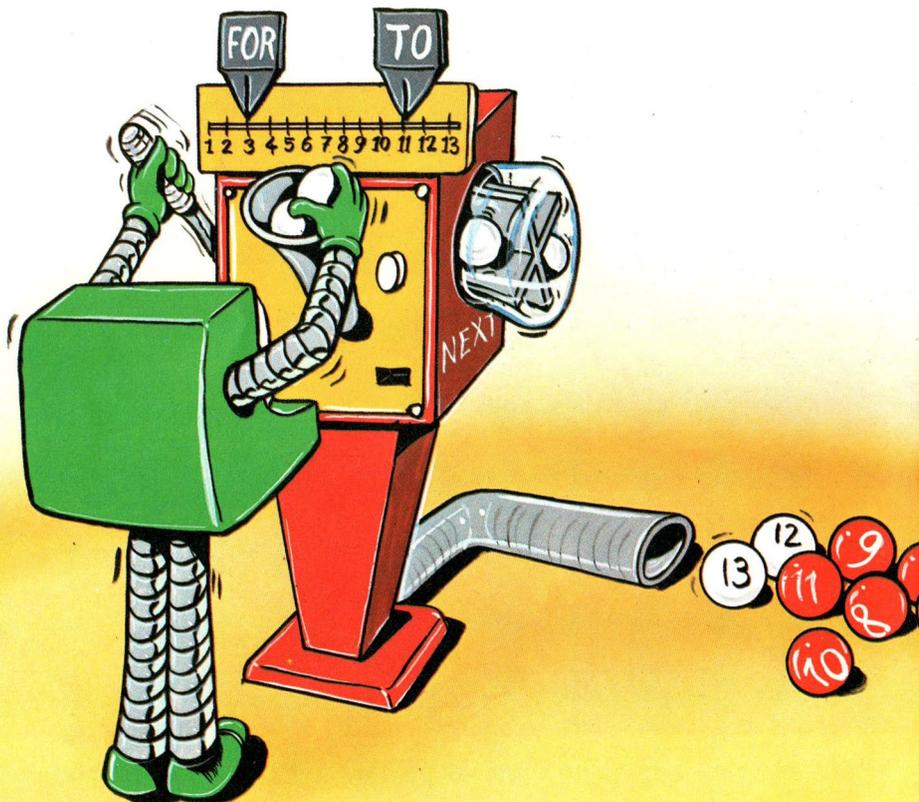
```
NEXT C
```

La prima volta che il ciclo viene eseguito C è posto uguale al valore di S. Si eseguono quindi tutte le istruzioni. Arrivati a NEXT il valore di C viene incrementato e confrontato automaticamente con A. Se C risulta minore di A, il ciclo viene nuovamente ripetuto, altrimenti si continua con l'istruzione che segue il NEXT.
La procedura prosegue pertanto finché il contatore C raggiunge il valore finale di A.
Riprendendo il problema della moltiplicazione si poteva allora dare questa soluzione:

```
10 FOR C = 1 TO 5
20 LET A  = A * C
30 PRINT A
40 NEXT C
```

C si chiama variabile di controllo del ciclo (o contatore) e può assumere un qualsiasi nome (legale) consentito. Nell'esempio avrai notato che 1 è il suo valore iniziale e 5 è il valore finale o di test.
Nell'istruzione FOR si possono utilizzare anche delle espressioni; per esempio:

```
FOR A = M + 5 TO B/5
```

oppure delle variabili, a patto che siano prima

state impostate. Ad esempio:

```
10 LET DA = 5 : LET A = 15
20 FOR C = DA TO A
30 ....
40 NEXT C
```

È inoltre possibile incrementare il valore della variabile contatore con un passo diverso da 1, semplicemente utilizzando la parola STEP (passo) seguita dal valore di cui si vuole incrementare ogni volta il contatore:

```
FOR I = 2 TO 10 STEP 2
```

I assumerà allora i valori 2, 4, 6, 8, 10.
È anche possibile scrivere programmi in cui i cicli contengano al proprio interno altri cicli:

```
10 FOR I=0 TO 10
   STEP 3
20 FOR J=3 TO 9
30 ......
40 ......
50 ......
60 NEXT J
70 NEXT I
```

Si dice allora che i cicli sono nidificati (ti ricordi di IF...THEN GOTO...). Naturalmente i contatori dei singoli cicli devono essere diversi. Il ciclo interno ad un altro ciclo deve inoltre essere completamente contenuto nel primo ciclo; non è quindi possibile sovrapporre parti di cicli. Ad esempio:

```
100 FOR I=13 TO 20
110 FOR J=0 TO 10
120 ......
130 ......
140 ......
150 ......
160 NEXT I
170 NEXT J
```

È sbagliato! I due cicli sono infatti sovrapposti. Molto pericoloso!
Se la variabile indice è inizialmente più grande del valore finale, il ciclo viene eseguito una sola volta. Ad esempio:

```
50 FOR I=20 TO 5
60 PRINT I
70 NEXT I
```

verrà praticamente ignorato.
Il passo del ciclo può essere anche negativo:

```
30 FOR X = 13 TO 10 STEP - 2
40 ......
50 ......
60 ......
70 NEXT X
```

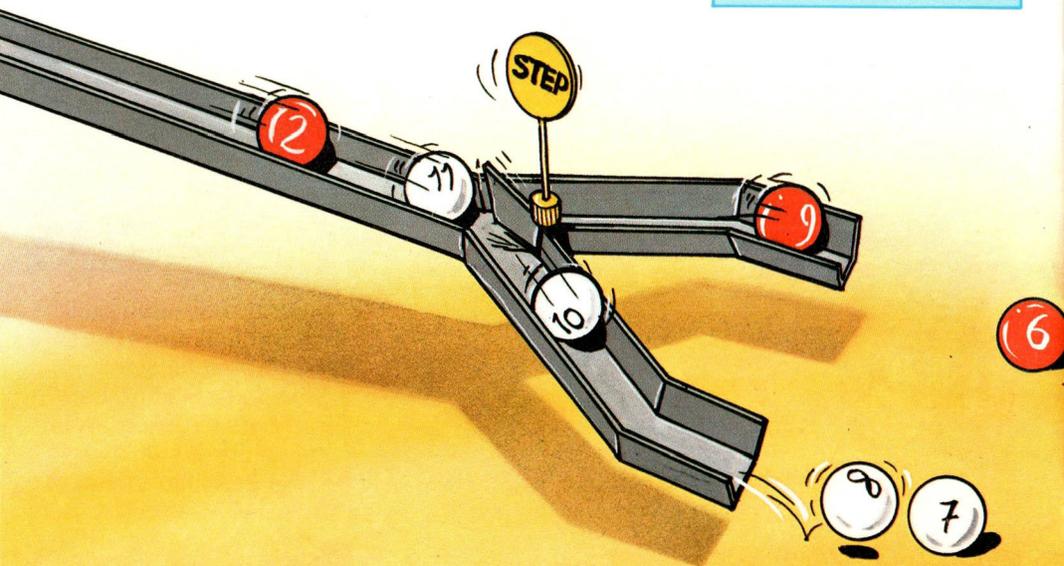
In questo caso il ciclo viene eseguito finché X, decrementandosi di due ogni volta, non diventa minore di 10.
Se il valore del passo viene posto uguale a 0, il ciclo si ripete indefinitamente:

```
10 FOR K = 1 TO 10 STEP 0
20 PRINT K
30 NEXT K
```

L'unico risultato di questo programma sarà quindi la continua visualizzazione sullo schermo del valore iniziale di K, cioè 0.
Devi inoltre fare attenzione al fatto che uscire dall'interno di un ciclo prima che la variabile di controllo abbia raggiunto il valore finale fa sì che il calcolatore aspetti la chiusura di un ciclo che non troverà mai. In certi casi si potrà arrivare addirittura a provocare messaggi del tipo:

```
NEXT WITHOUT FOR
```

oppure

```
VARIABLE NOT FOUND
```

È pertanto buona pratica evitare di inserire in un ciclo delle istruzioni di salto (cioè dei GOTO); ne trarrà giovamento anche la leggibilità del programma.
Un altro errore molto comune è quello di utilizzare un numero di NEXT superiore a quello dei FOR.
In questo caso il tuo Spectrum risponderà con:

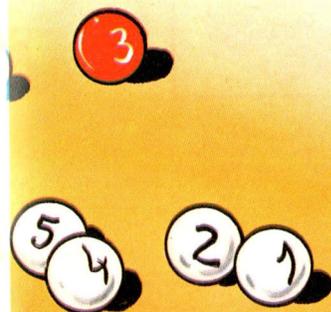
```
NEXT WITHOUT FOR
```

Un'ultima raccomandazione: il nome della variabile di controllo deve sempre essere formato da una sola lettera, come i, j, k, x ...

Sintassi del comando

FOR indice = valore iniziale TO valore finale [STEP passo]
NEXT indice

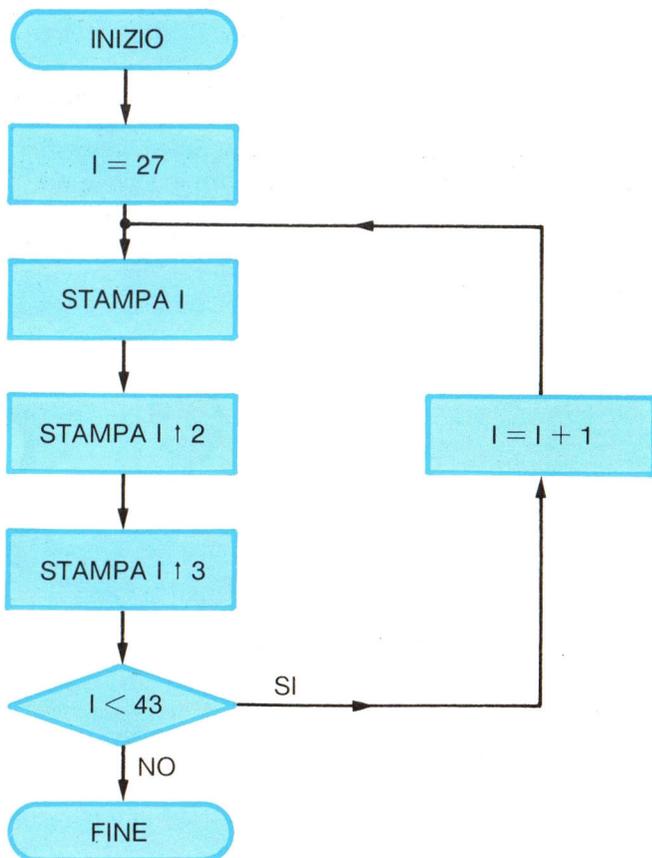
---

dove:
- indice è il nome di una variabile numerica usata come contatore
- valore iniziale è il valore di partenza di indice
- valore finale è il valore che l'indice deve uguagliare o superare (test)
- passo è l'incremento che ad ogni iterazione subisce l'indice. Se non è specificato, viene posto pari ad 1 e può essere negativo.

I cicli automatici

Si chiamano cicli automatici tutti quei cicli che eseguono ripetutamente una sequenza di istruzioni facendo uso dell'istruzione FOR...NEXT.
Nella maggior parte dei programmi i cicli automatici sono di impiego talmente utile e frequente da costituire un importantissimo strumento nelle mani di qualsiasi programmatore.
Gli esempi applicativi che seguono ti aiuteranno quindi a chiarire ed approfondire i concetti teorici che già conosci sia riguardo all'uso dei cicli che del comando FOR ... NEXT.

Quadrati e cubi

Come primo esempio consideriamo questo problema: trovare i quadrati ed i cubi dei numeri compresi tra 27 e 43.
Una possibile risoluzione potrebbe essere:

A questo schema a blocchi corrispondono i due seguenti programmi BASIC.
Il primo fa uso di un ciclo controllato (creato cioè "artificialmente" dal programmatore); il secondo di un ciclo automatico:

```
10 LET I = 27
20 PRINT I↑2,
30 PRINT I↑3
40 IF I<43 THEN I=I+1:GOTO20
50 REM FINE
```

```
10 FOR I=27 TO 43
20 PRINT I↑2,
30 PRINT I↑3
40 NEXT I
50 REM FINE
```

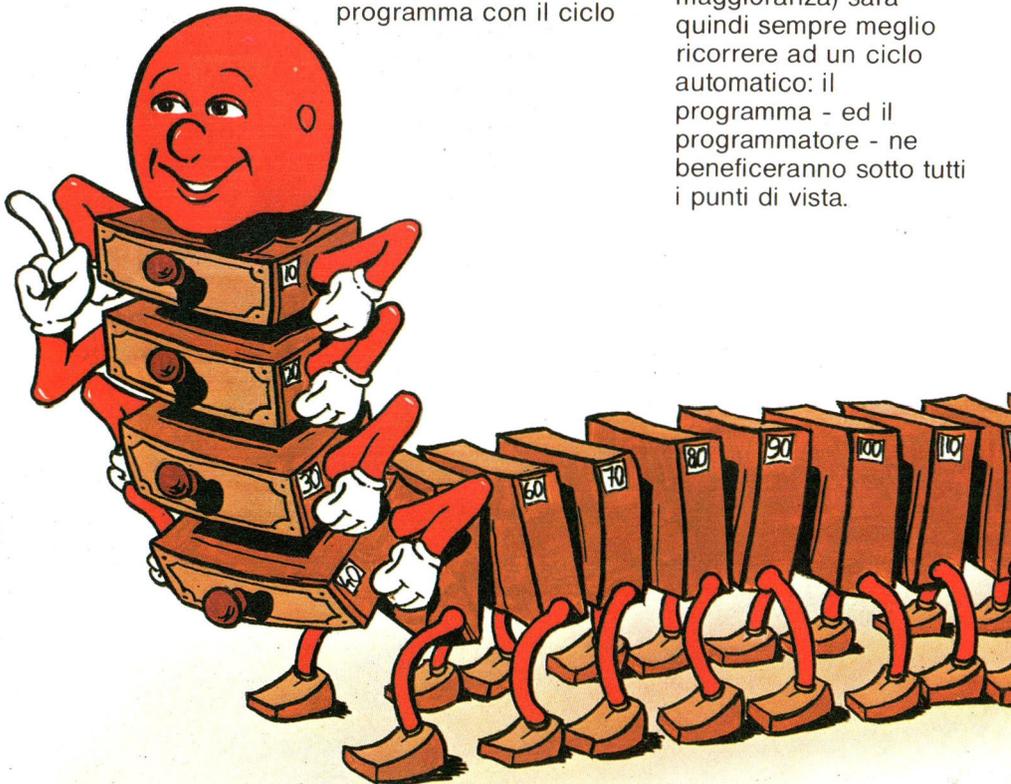
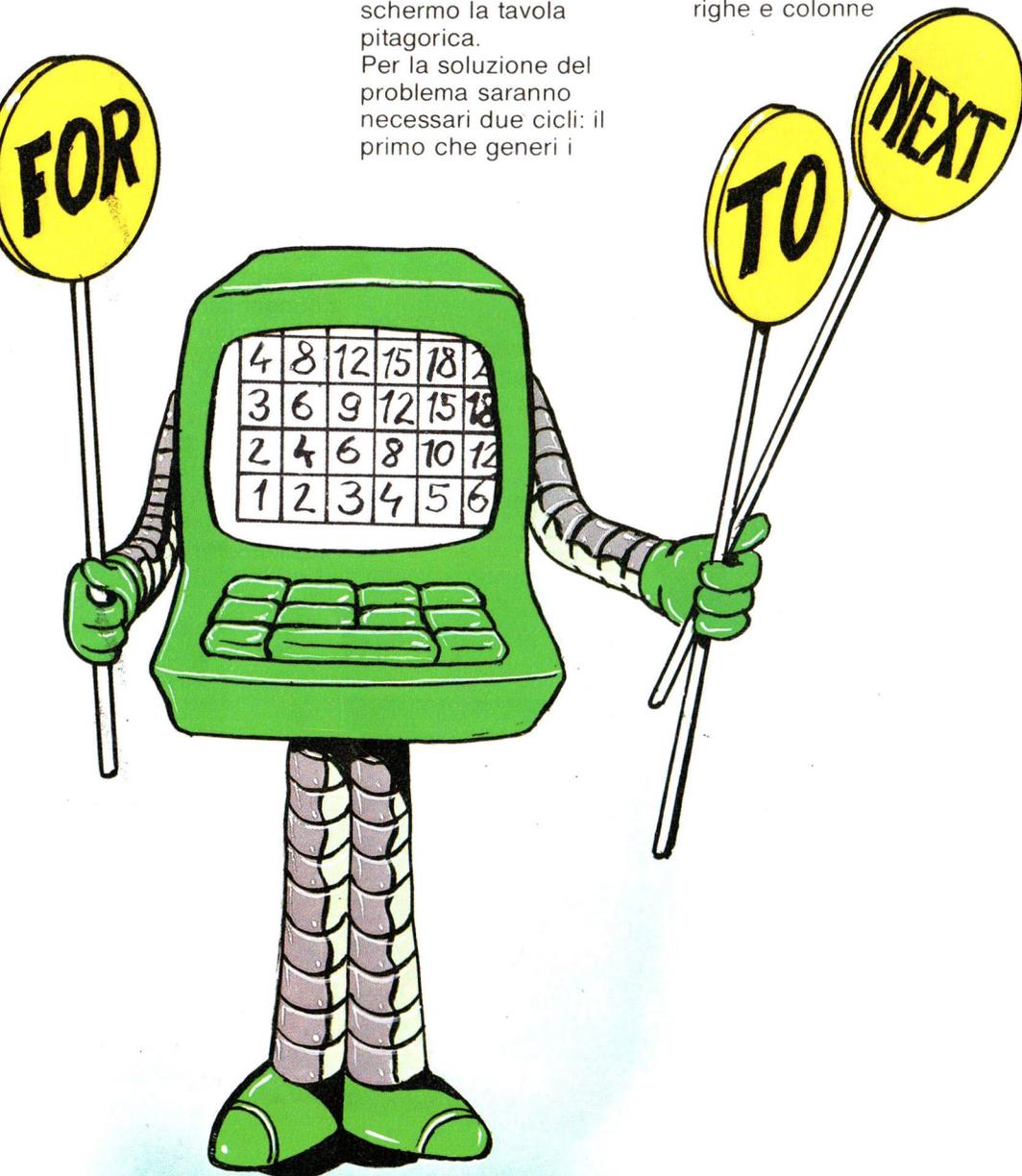
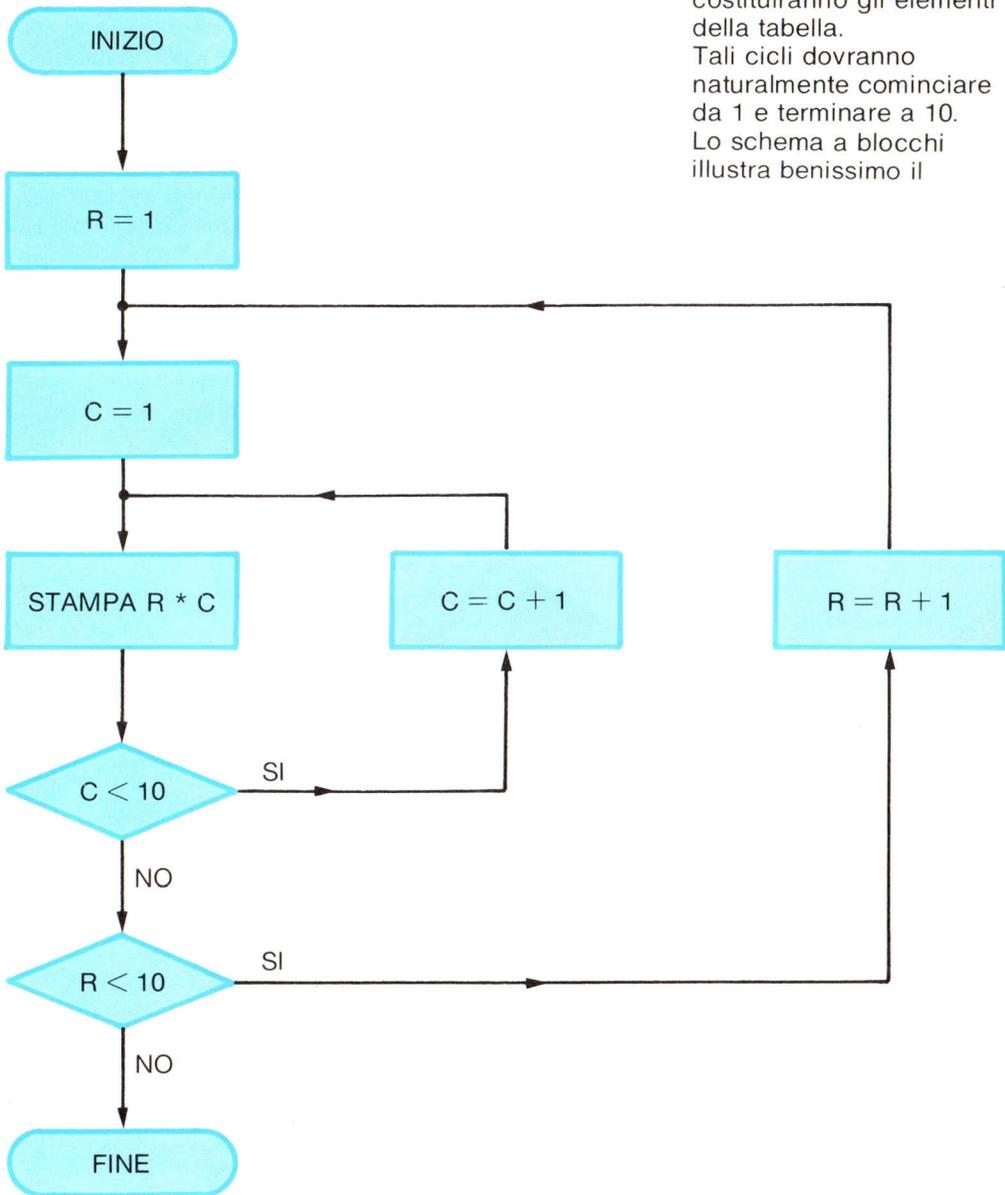
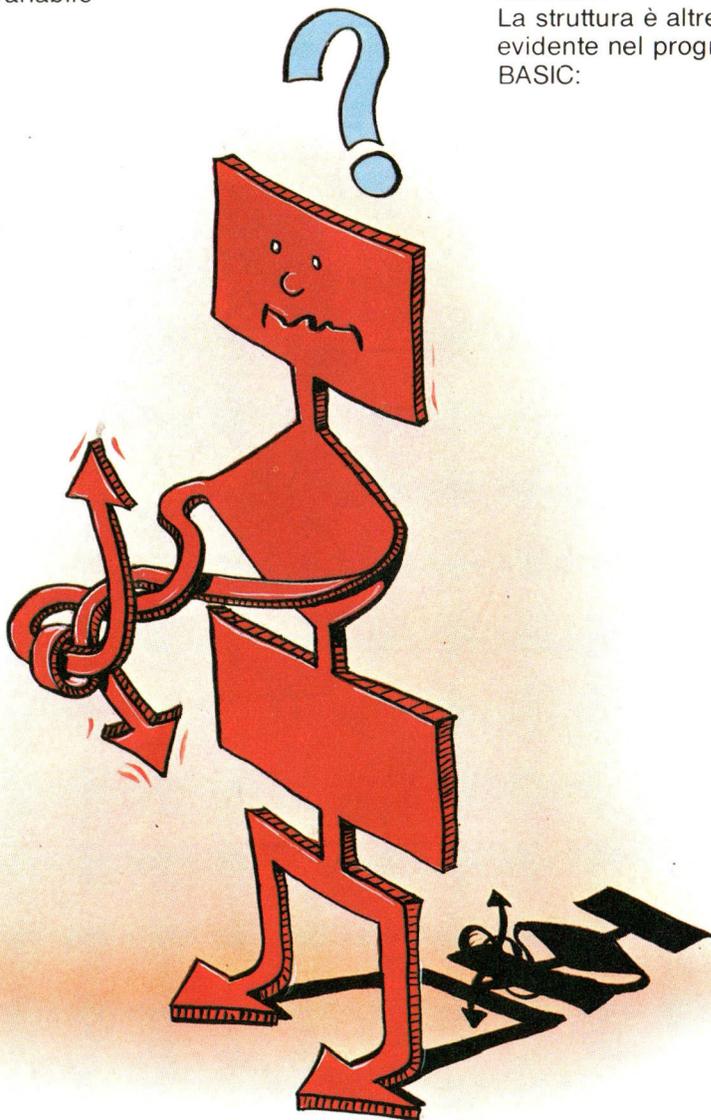
Al lettore balza all'occhio immediatamente la migliore leggibilità del programma con il ciclo automatico: in esso si distinguono subito l'inizio e la fine del ciclo, cosa che invece non accade nel primo listato. Anche per il tuo Spectrum la seconda soluzione è preferibile: l'esecuzione del ciclo è infatti affidata ad una istruzione esplicitamente concepita per risolvere questo tipo di problemi e quindi ad esso più congeniale.
Nei casi in cui è possibile (e sono la maggioranza) sarà quindi sempre meglio ricorrere ad un ciclo automatico: il programma - ed il programmatore - ne beneficeranno sotto tutti i punti di vista.

PROGRAMMAZIONE

Tavola pitagorica

Come secondo esempio scriviamo un programma che visualizzi sullo schermo la tavola pitagorica.
Per la soluzione del problema saranno necessari due cicli: il primo che generi i numeri corrispondenti alle righe, il secondo alle colonne. I prodotti tra righe e colonne

costituiranno gli elementi della tabella.
Tali cicli dovranno naturalmente cominciare da 1 e terminare a 10.
Lo schema a blocchi illustra benissimo il

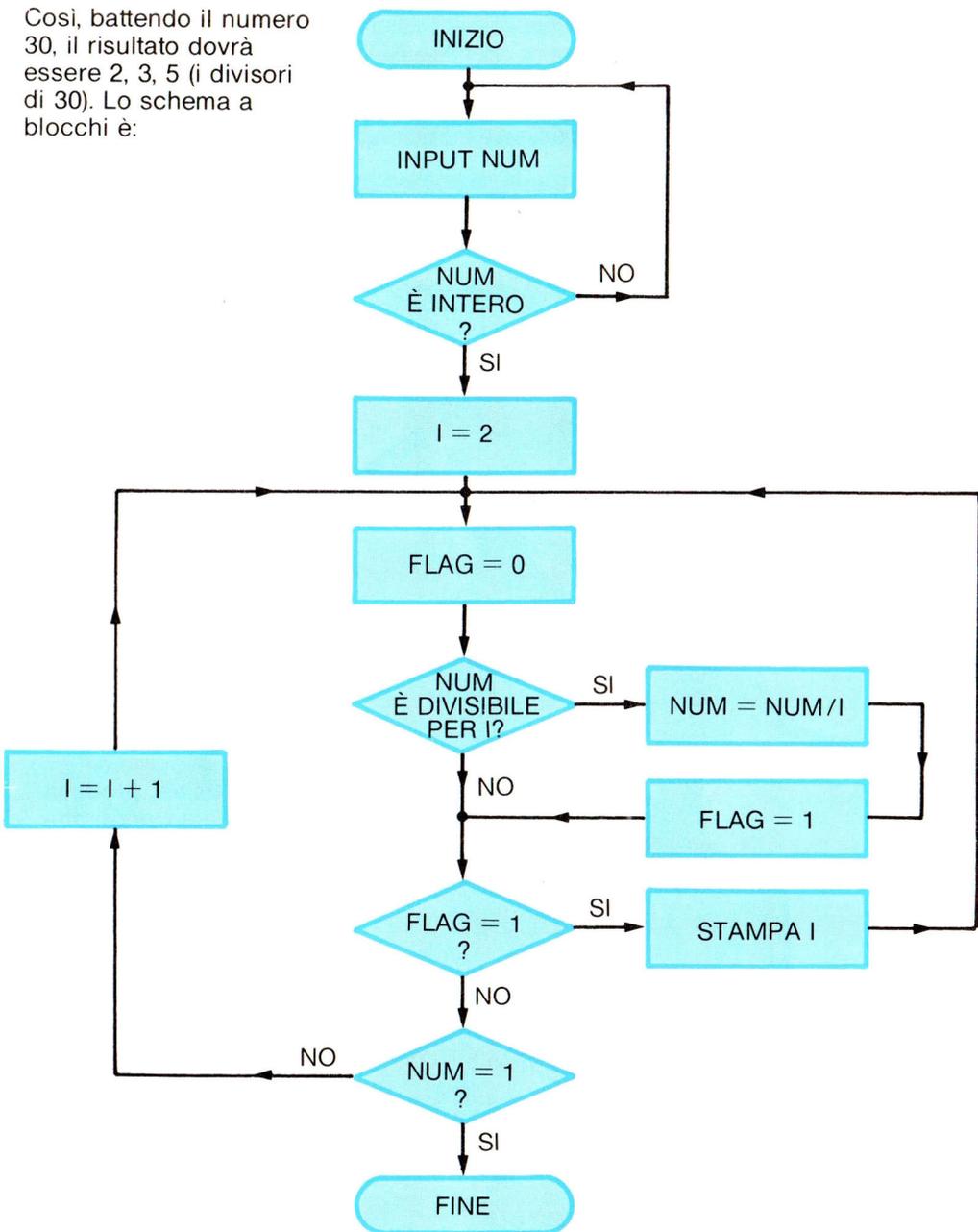
concetto di "cicli nidificati"; il ciclo che modifica ed incrementa la variabile C è infatti completamente inserito in quello della variabile R.

Sul tuo Spectrum i cicli FOR possono essere nidificati, cioè posti l'uno all'interno dell'altro, fino ad un massimo di 10 volte. Non abusarne però, perché diventa già molto difficile seguire il flusso di più di tre cicli nidificati.
La struttura è altrettanto evidente nel programma BASIC:

```
10 FOR R = 1 TO 10
20 FOR C=1 TO 10
30 PRINT R * C; " ";
40 IF R * C < 10 THEN PRINT " "; : REM se il
   prodotto è composto da una cifra sola, allora
   stampa uno spazio
50 NEXT C
60 PRINT
70 NEXT R
80 REM FINE
```

Le righe 40 e 60 sono state incluse nel programma per allineare le tabelline, inserendo opportuni spazi tra le varie righe e colonne. Per capire quale sia il loro effetto prova ad eliminarle (comincia con la linea 40 e quindi con la 60): da ciò che comparirà sullo schermo dovresti essere immediatamente in grado di comprenderne la funzione e l'efficacia.

Scomposizione in fattori primi

Come ultimo esempio vediamo infine il seguente problema. Scrivere un programma che, accettato in ingresso un numero intero qualsiasi, produca in uscita tutti i valori che di tale numero sono divisori primi (ricordati che i numeri primi sono quei numeri divisibili solo per 1 e per se stessi).

Così, battendo il numero 30, il risultato dovrà essere 2, 3, 5 (i divisori di 30). Lo schema a blocchi è:

Ed il corrispondente listato BASIC:

```
 5 INPUT "NUMERO =" ; NUM
10 CLS : PRINT "FATTORI PRIMI DI" ; NUM
20 IF NUM <> INT(NUM)GOTO 5: REM NUM È UN NUMERO INTERO?
30 FOR I = 2 TO NUM
40 LET FLAG = 0:REM FLAG È <> 0 QUANDO NUM È DIVISIBILE PER I
50 IF NUM/I = INT (NUM/I) THEN LET NUM = NUM/I : LET FLAG = 1
60 IF FLAG = 1 THEN PRINT I: GOTO 40
70 IF NUM = 1 THEN GOTO 90
80 NEXT I
90 REM FINE
```

Il programma comincia verificando che il valore del numero battuto sulla tastiera non abbia cifre decimali; in caso contrario ne domanda in ingresso uno nuovo.
Dalla linea 30 comincia la vera e propria fase di esecuzione e risoluzione del problema: se NUM (il numero battuto in ingresso) è divisibile per I (linea 50), allora FLAG assume valore 1.
FLAG è una variabile adibita ad indicatore: quando assume valore 1 significa che I è un divisore di NUM e quindi va stampato. Se invece vale 0, I non è divisore di NUM e bisogna incrementarlo di uno.
Man mano che il ciclo prosegue NUM diventa sempre più piccolo; alla fine assumerà valore 1.
A quel punto il problema sarà risolto: sullo schermo saranno infatti comparsi tutti quei numeri che, moltiplicati tra loro, formavano il valore iniziale di NUM.

VIDEOESERCIZI

Aiutandoti con la funzione CODE, metti in ordine crescente di codice le seguenti stringhe:

"BIT",

"SUPER SINC",

"HOME COMPUTER",

CHR$ (13) + "VIDEOG",

"VIDEOBASIC"

CODICE	STRINGA

Aiutandoti con la tabella ASCII, prevedi e scrivi l'output del seguente programma:

```
10 PRINT CHR$ (86); CHR$ (73); CHR$ (68); CHR$ (69);
20 PRINT CHR$ (79); CHR$ (66); CHR$ (65);
30 PRINT CHR$ (83); CHR$ (73); CHR$ (67)
```


A) cronometra e annota la durata del programma.
B) cerca, annota poi elimina la linea di ritardo
C) cronometra nuovamente e annota il tempo.

```
10 CLS
20 FOR D = 10 TO 90 STEP 8
30 PRINT "DATO"; D
40 FOR P = 1 TO 3000: NEXT P
50 NEXT D
```

A	TEMPO
B	N° LINEA DI RITARDO
C	TEMPO

Sostituisci ora la linea 40 del precedente programma con la seguente:

```
40 PAUSE 150
```

Cronometra nuovamente e ricerca un valore di pausa analogo a quello del LOOP: FOR P = 1 TO 3000 : NEXT P.